REMARKS

In the application, Claims 1-17 are pending and rejected. After due consideration of the Examiner's comments, the application has been amended as set forth above. In view of these amendments, Applicant submits that the claims are allowable over the prior art and requests that the Examiner issue a notice of allowance.

Rejections under 35 U.S.C. §112

The Examiner rejects claims 10 and 11 under 35 U.S.C. §112, 2nd para. as being indefinite. Specifically, the Examiner asserts that the term "dirty" is a relative term that renders the claim indefinite.

Applicant respectfully submits that the term "dirty" when used to describe data is not indefinite in that it would be recognized by those in the art as meaning incomplete, corrupted or other data that cannot be properly processed by a learning machine. Such a definition is provided in Pat. No. 6,157,921, column 8, line 13. This patent is incorporated in the present application by reference at page 7, line 20. See also the article by Hernandez and Stolfo entitled "Real-world Data is Dirty: Data Cleansing and The Merge/Purge Problem", *Data Mining and Knowledge Discovery*, Vol. 2, Issue 1, Jan. 1998, pp. 9-37. A copy of the abstract for this article is attached as Exhibit A.

The Examiner rejects claim 17 under 35 U.S.C. §112, 2nd para. as being indefinite. Specifically, the Examiner asserts that the dichotomy is not completed.

The claim has been amended to address this rejection.

Rejections under 35 U.S.C. §102

The Examiner rejects claims 1-17 under 35 U.S.C. §102(a) as being anticipated by Cristianini and Shawe-Taylor (*An Introduction to Support Vector Machines*, Cambridge University Press, 2000) (hereinafter referred to as "CST").

The independent claims have been amended to include the limitation that the probability of the two classes be the same under the distribution given by the first eigenvector. This constraint is imposed by the second eigenvector for the purpose of maximizing the alignment.

Additional text has been added to the specification to clarify this point. The added text does not constitute new matter in that it is taken from provisional application Serial No. 60/272,391, to which this application claims priority and which has been incorporated by reference into this application. A copy of pages 9 and 10 of the provisional application with the relevant portion highlighted is attached hereto as Exhibit B.

The Examiner alleges that claim 7, which in the originally-submitted claims first introduces the use of the second eigenvector to obtain alignment, is anticipated by CST, p. 156, lines 29-41. Applicant respectfully submits that the cited paragraph neither mentions alignment nor describes a concept similar to alignment. Rather, the paragraph discusses maximal margin classifiers and soft margin classifiers which are merely different forms of analysis that can be formed by support vector machines. Maximal margin classifiers are used with linearly separable data where there is no overlapping of the samples, where the classifier with the largest margin will provide the lowest risk. Soft margin classifiers have some overlapping classes and are effected by using slack variables that permit some variables to be misclassified. These classifiers have nothing to do with alignment, which is used to determine similarity within and between the clusterings of a set of points. Quoting from the specification at page 12, line 6, "Alignment captures the notion of a good clustering as achieving high similarity within the clusters and low similarity between them;" and at page 16, line 20, "This approach is directed to finding clusters that have minimal "scatter" around their mean. Among other appealing properties of the alignment is that this quantity is sharply concentrated around its mean." See also the discussion on page 4 of the publication by Marina Meilă, "Data Centering in Feature Space", from the 9th International Workshop on Artificial Intelligence and Statistics, Jan. 2003, attached hereto as Exhibit C.

CST does not disclose kernel alignment. Further, CST does not disclose that the second eigenvector is used to select the best aligned kernels by imposing the constraint that the probability of the two classes be the same under the distribution given by the first eigenvector. Therefore, CST fails to disclose each and every element of the claimed invention as required under §102. Accordingly, Applicant respectfully requests that the prior art rejection be withdrawn.

Should the Examiner believe that prosecution of this application might be expedited by further discussion of the issues, he is invited to telephone the undersigned attorney for Applicant at the telephone number indicated below.

Respectfully submitted,

Dated: January 27, 2005

By: ______________________
Eleanor M. Musick
Attorney for Applicants
Registration No. 35,623

PROCOPIO CORY HARGREAVES & SAVITCH LLP
530 B Street
Suite 2100
San Diego, California 92101-4469
Telephone: (760) 931-9703 (direct)
Facsimile: (760) 931-1155
emm@procopio.com

Docket No. 0171 [112092-000041]

Real-world Data is Dirty: Data Cleansing and The Merge/Purge Problem *

Mauricio A. Hernández[†] mauricio@cs.columbia.edu

Salvatore J. Stolfo sal@cs.columbia.edu

Department of Computer Science
Columbia University
New York, NY 10027

Abstract

The problem of merging multiple databases of information about common entities is frequently encountered in KDD and decision support applications in large commercial and government organizations. The problem we study is often called the Merge/Purge problem and is difficult to solve both in scale and accuracy. Large repositories of data typically have numerous duplicate information entries about the same entities that are difficult to cull together without an intelligent "equational theory" that identifies equivalent items by a complex, domain-dependent matching process. We have developed a system for accomplishing this Data Cleansing task and demonstrate its use for cleansing lists of names of potential customers in a direct marketing-type application. Our results for statistically generated data are shown to be accurate and effective when processing the data multiple times using different keys for sorting on each successive pass. Combing results of individual passes using transitive closure over the independent results, produces far more accurate results at lower cost. The system provides a rule programming module that is easy to program and quite good at finding duplicates especially in an environment with massive amounts of data. This paper details improvements in our system, and reports on the successful implementation for a "real-world" database that conclusively validates our results previously achieved for statistically generated data.

Keywords: data cleaning, data cleansing, duplicate elimination, semantic integration

*This work has been supported in part by the New York State Science and Technology Foundation through the Center for Advanced Technology in Telecommunications at Polytechnic University, by NSF under grant IRI-94-13847, and by Citicorp.

[†] This author is now at the University of Illinois at Springfied. Work at Columbia University was supported by an AT&T Cooperative Research Program Fellowship.

1998
Data Mining and Knowledge Discovery
v. 2, I1 (Jan. 98) p. 9-37

Theorem 2 Similarly can be done for 'leave one out' cases: if, by removing a variable altogether, the function does not change much, then it i concentrated.

NOTICE: SUP. OF SUMS IS CONCENTRATED !!!!!!

Consequence: concentration of spectrum

This means that we can trust the results from spectral graph theory to give us good hypotheses ... [CHECK THIS AGAIN !]

We could also use Andre's approach, with stability analysis for producing bounds ...

5 Spectral Kernel Algorithms

We will assume that we are given a dataset of m points drawn from a set X according to a fixed distribution $\mathcal{D}$. In the supervised learning case we will also assume we are given a vector $y \in \{-1, +1\}^m$ (labels), and in the semisupervised or transductive case that we are given a vector $y \subset \{-1, *, +1\}^m$, where $*$ mans that there is no label for a particular point.

A Distribution on the Data Set Consider a graph G whose adjacency matrix is K. Consider a random walk on the graph. Consider its stationary or ergodic distribution. It somehow measures the popularity of a given node, how much time a random walker will spend on it. This depends on the number, popularity and closeness of its neighbours.

It is given by the first eigenvector of K.

The distance between the first 2 eigenvalues gives information about the degree of connectivity of the graph. It can be used to measure the amount of structure in the data.

Similarly, the entropy of the stationary distribution, can quantify it.

NOTICE: lifting can be used to explicitely adapt the first eigenvector, and hence adapt kernels.

Remark One can perform data cleaning by considering the label as another feature. In this way, spotting isolated points would amount to spotting unusual combinations of x and y.

Maximum Alignment In [5] we proposed the quantity $A = y'Ky$ (called 'alignment') to measure the level of fitness between a kernel and a fixed labeling of the data. The purpose was to better choose kernel parameters. An alternative - and perhaps more interesting - problem is to select the best aligned set of labels (possibly subject to some constraint). `[can it be NP complete ? reduces to bipartitioning if replaces K with L]`

This is an absolute measure of a kernel: its second eigenvalue. It can be used to optimize kernel parameters.

We can look at an approximation of it, with eigentechniques: its value can be lower-bounded with the second eigenvalue.

Consider the constraint C1.: $\sum_+ p_i = \sum_- p_i$. We relax for the moment the requirement that $y = -1$ or $+1$, and we ask that $\sum y_i^2 = m$ and (we will see later why) $\sum_+ p_i = \sum_- p_i$, that is the two classes have equal probability under the distribution of the first eigenvector. [DISCUSS THIS !]

Under these constraints, the alignment can be maximized by spectral techniques.

Defining $v = \frac{y}{\sqrt{(m)}}$, the problem becomes:

$$min_{C1, \sum y_i^2 - m} y' K y = min_{\sum v_i^2 = 1; + otherconstraint} \frac{v' K v}{v' v} = \lambda_2$$

where v_2 is the minimizer, the second eigenvector.

That is the second eigenvector maximizes the alignment under the constraint, and thresholding it provides with a labelling that approximately maximizes alignment. The second eigenvalue gives the value of the alignment in this case, and is a lower bound under the true value of the optimal alignment.

A very important remark. Now one has an 'absolute' measure of kernel alignment: the maximal alignment achieved on all possible labelings. And this is equivalent to the second eigenvalue. So one can also tune the kernel parameters in a principled way, maybe by gradient descent, so to achieve maximal alignment. The thresholding of the second eigenvector will then be the output.

The constraint is not however just a technical problem: it actually has a very clear meaning: IT REQUIRES THE PROBABILITY OF THE TWO CLASSES TO BE THE SAME UNDER THE DISTRIBUTION GIVEN BY THE FIRST EIGENVECTOR... (analogous to the requirement of a balanced split in the laplacian case. how to relax it ?)

This gives us many natural algorithms:

Unsupervised - clustering The choice of a kernel automatically defines two classes in the unlabeled dataset by means of the sign of the second eigenvector. Successive eigenvectors can be used for further partitionings. A measure of the goodness of a given cluster is the second eigenvalue, or its aligment. Kernel parameters can be tuned to optimize it. Analogously can be done with the laplacian.

The experiments describe the performance of the clustering algorithm tested on Wisconsin Breast Cancer and on Ionosphere data. In the experiments, the algorithm was used to split in two classe the data, then the split was compared with the labels, to measure the error.

One can take also advantage of the information contained in the other kernels, as can be seen in the other plot.

An algorithm for assessing the clustering power of a kernel goes at follows:

- build K
- build L
- compute eig$L$
- number of clusters is number of (approx) zero eigenvalues

For multiclass cases, the alignment is defined, but one should replace $y_i y_j$ with $y_i == y_j$ in the matrix to be compared vs K. Again, it would give a

Data Centering in Feature Space

Marina Meilă
Department of Statistics
University of Washington
Seattle, WA 98195-4322
mmp@stat.washington.edu

Abstract

This paper presents a family of methods for data translation in feature space, to be used in conjunction with kernel machines. The translations are performed using only kernel evaluations in input space. We use the methods to improve the numerical properties of kernel machines. Experiments with synthetic and real data demonstrate the effectivenes of data centering and highlight other interesting aspects of translation in feature space.

1 Introduction

Support vector machines (SVMs for short) classify data by mapping it into a high (possibly infinite) dimensional *feature* space and constructing a maximum margin hyperplane to separate the classes in that space. Operations in the feature space are rendered independent of its dimension by what is commonly called now the "kernel trick", the use of an efficiently computable *kernel function* for the scalar product in feature space.

The SVM classifier is learned from data by means of the *Gram matrix* K consisting of the pairwise scalar products of the data points in feature space. If, in the feature space, the origin is far away from the convex hull of the data, then the elements of K have about the same value and, as a result, the matrix K is ill-conditioned. Figure 1 illustrates such a situation, showing that the performance of the resulting classifier degrades.

The present work sets out to correct this problem, by shifting the data such that the origin is located in the convex hull of the data. While this is almost trivial for a linear classifier, it is not so for non-linear SVMs where the data are mapped non-linearly into a high-dimensional space that is not explicitly represented. Thus, the challenge is to perform the shift and to compute the resulting SVM using only "allowed" operations, that is applying the kernel to points in the input space.

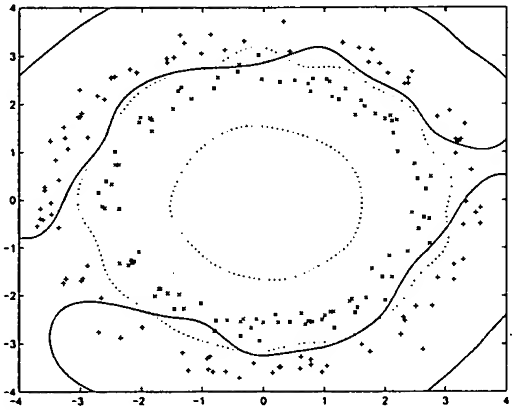

Figure 1: Original SVM classifier (dotted line) and classifier obtained after translating the origin in feature space by ≈ 500 units (full line). The two classes are represented by "+" and "x" respectively. The kernel used is the RBF kernel.

This paper presents kernel tricks that allow one to perform a large variety of origin translations in the feature space of a non-linear kernel machine. In its simplest version, this method of data centering in feature space has been in use for a long time (see e.g [4]). Here we show that one can formulate data centering in the form of criterion to be optimized by a translation in feature space, and that this translation can be performed entirely by "allowed" kernel operations.

There has been previous work on adapting kernels to the data, notably that of [1, 5, 6]. The current idea differs from the above in that it does not affect the geometry of the problem, it only affects its translation into a numerical task.

An origin placed far away from the data is not the only cause of numerical problems in SVMs. Another common problem is the inappropriate choice of kernel width (typical for RBF kernels) which can result into overfitting/underfitting [8]. A related problem known as "ridge effect" occurs in string kernels [13]: the data are almost or-

9TH International Workshop on Artificial Intelligence and Statistics, Jan. 2003 thogonal to each other in feature space. This can lead to both numerical problems and overfitting. While a translation of the data may alleviate the numerical problems, the geometry of the classifier cannot be influenced by origin shift in feature space. Therefore we will not deal with these problems in the current paper.

We start with a short introduction to support vector machines, then we introduce a simple method of data centering in section 3 and explain how to perform classification with shifted support vectors in section 4. Next we present a general method of shifting in feature space in order to optimize some given centering criterion. Experiments are presented in section 6 and the discussion in section 7 concludes the paper.

2 Support Vector Machines

We start by briefly introducing the SVM; for more details the reader is invited to consult [8]. In a classification task, we are given a set of n *data points* $\{\tilde{x}_1, \tilde{x}_2, \ldots \tilde{x}_n\}$, elements of an *input space* $\tilde{\mathcal{X}}$. Each data point $\tilde{x}_i$ is labeled by $y_i \in \{\pm 1\}$. The task is to use the data and labels to construct a classifier, i.e. to find a function f that predicts the label y of a new point $\tilde{x}$.

Input space and feature space. We call the space of the original data the *input space*. The data are mapped into a space $\mathcal{X}$ called the *feature space* by a function ϕ

$$x = \phi(\tilde{x}) \quad \text{for all } \tilde{x} \in \tilde{\mathcal{X}} \tag{1}$$

We assume that the data is *linearly separable* in $\mathcal{X}$, meaning that a hyperplane that separates the two classes exists. The feature space is a Hilbert space whose dimension d is commonly much larger than n the number of data points and it can be infinity (e.g in the *RBF* kernel [8]). The input space need not be a Hilbert space, it can be any set. The trick that makes SVMs work is never to explicitly represent points in feature space or ϕ itself. The SVM only makes use of scalar products of points in feature space, which are computed by the function

$$K(\tilde{x}, \tilde{z}) \equiv < \phi(\tilde{x}), \phi(\tilde{z}) > = < x, z > \tag{2}$$

called the *kernel* associated with ϕ. It is assumed that $K(\tilde{x}, \tilde{z})$ can be computed efficiently for any pair of inputs. To be represent a scalar product, a symmetric kernel K is subject to the Mercer condition [8], namely that it induces a positive definite integral operator on $\tilde{\mathcal{X}}$.

Finding the optimal hyperplane. The separating hyperplane is described by the equation $< w, x > +b = 0$, with w a vector in feature space and b a real number. The optimal hyperplane is found by solving an optimization problem in the variables $\alpha_i, \ i = 1, 2, \ldots n$.

$$\max_x \mathcal{V}(\alpha) \quad \text{s.t.} \quad \alpha_i \geq 0 \text{ and } \sum_{i=1}^{n} \alpha_i y_i = 0 \tag{3}$$

with

$$\mathcal{V}(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j < x_i, x_j > \tag{4}$$

The optimal hyperplane is then obtained from α_i by

$$w = \sum_{i=1}^{n} \alpha_i y_i x_i \tag{5}$$

$$b = y_i - \sum_{j=1}^{n} \alpha_j y_j < x_i, x_j >, \text{ for some } i \text{ s.t } \alpha_i \neq 0 \tag{6}$$

Note that although the optimization problem involves the data images in feature space, the data points enter $\mathcal{V}$ only via the pairwise scalar products $< x_i, x_j >= K(\tilde{x}_i, \tilde{x}_j)$ which can be computed using the kernel function. This is the celebrated "kernel trick" of support vector machines. The matrix

$$K = [K(\tilde{x}_i, \tilde{x}_j)]_{i,j=1,\ldots n} \tag{7}$$

is called the *Gram matrix*[1]. Throughout the rest of the paper, we assume that a function SVM-SOLVER is given. The function SVM-SOLVER takes as input a Gram matrix K and a set of labels y returns the parameters $b, \ \alpha_i, \ i = 1, \ldots n$ of an SVM.

Classifying with SVMs. When a new point $\tilde{x}$ is presented for classification, its label is computed by

$$y = f(\tilde{x}) = \text{sign}\left[\sum_i \alpha_i y_i K(\tilde{x}, \tilde{x}_i) + b\right] \tag{8}$$

Note that the function f uses only kernel computations so it can be computed explicitly. If the kernel K is a non-linear function in each argument, then the resulting classifier f is a non-linear classifier.

SVM extensions For the sake of simplicity, we have presented here only the most basic version of SVM. Many other version exist that build upon the basics, some meant to deal with the case of non linearly separable data (C-SVM [3], ν-SVM [11]), others adapted for classfication from positive examples only [9], and others meant to deal with more than two classes [10]. All SVM versions cited here have in common the use of the Gram matrix as the only vehicle by which the data enter the SVM training. Therefore, the methods for data translation we present here should apply to them as well.

[1] We shall use the same notation K for both the Gram matrix and the kernel; the distinction will be evident from the context.

3 A simple centering method

As shown in section 1, if in feature space the origin lies far away from the data, then the matrix K will have almost equal elements and will be ill-conditioned.

How can we establish if, in the high-dimensional feature space, the origin lies "between" the classes or far-away from them? One way is to look at $K(\tilde{x}_i, \tilde{x}_j)$ when data points i, j belong to different classes. If this scalar product is negative, it means that the points are seen from the origin under an obtuse angle, in other words the origin lies approximately between the two points. If we denote by K_a the kernel representing the scalar product with the origin shifted in a

$$K_a(\tilde{x}, \tilde{z}) \;\equiv\; < x, z >_a \;\triangleq\; < x - a, z - a > \qquad (9)$$

then we can define the optimal position of the origin to be the location a that minimizes

$$J(a) \;=\; \sum_{y_i=1} \sum_{y_j=-1} K_a(\tilde{x}_i, \tilde{x}_j) \qquad (10)$$

Using

$$< x, z >_a \,=\, < x, z > - < x, a > - < z, a > + < a, a > \qquad (11)$$

and letting $n^+(n^-)$ denote the number of data points with $+1(-1)$ labels, we can rewrite $J(a)$ as a quadratic criterion in a whose (unique) minimum is at

$$a \;=\; \frac{1}{2n^+} \sum_{y_i=1} x_i + \frac{1}{2n^-} \sum_{y_i=-1} x_i \qquad (12)$$

Thus the optimal a according to (10) is positioned halfway between the centers of gravity of the two classes in feature space. The kernel K_a for this position of the origin may not be computable in closed form; nevertheless, we can obtain the Gram matrix K_a necessary to solve the SVM optimization problem using only calls to the original kernel K. This is a consequence of the fact that a is a linear combination of data points in feature space. Denote

$$\gamma_i \;=\; \begin{cases} (2n^+)^{-1}, & y_i = 1 \\ (2n^-)^{-1}, & y_i = -1 \end{cases} \qquad (13)$$

and $\bar{\gamma} \;=\; [\gamma_1 \;\ldots\; \gamma_n]^T, \Gamma = [\bar{\gamma} \ldots \bar{\gamma}]$. Then the "centered" Gram matrix is given by

$$K_a \;\equiv\; [K_a(\tilde{x}_i, \tilde{x}_j)]_{ij} \;=\; K - \Gamma^T K - K\Gamma + \Gamma^T K \Gamma \qquad (14)$$

Having obtained K_a, a call to SVM-SOLVER(K_a, y) will output the parameters b, α_i, $i = 1, \ldots n$ of an SVM.

The optimal a obtained by the centering as in (12) may not belong to the data manifold in feature space, hence it is generally not representable by a point $\tilde{a}$ in input space. In the next section we show that this fact does not preclude us us classifying new data points with the centered kernel.

The criterion (10) and its solution can be generalized to problems that involve more than two classes. The details are presented in the longer version of the paper [7] [2].

Both the criterion (10) and its multiclass extension guarantee that the origin is contained within the convex hull of the data after the shift. They are very similar to the "standard" data centering method (e.g [4]) which moves the origin at the center of gravity of the data. In terms of the γ_i coefficients above, movig the origin to the center of gravity of all the data amounts to setting $\gamma_i = \frac{1}{n}$.

4 SVM classification with centered data

We explain now how to perform classification with centered data. Denote by b, α_i the output of SVM-SOLVER(K_a, y). According to equation (8), classifying a new data point $\tilde{x}$ is done by

$$f_a(\tilde{x}) \;=\; \text{sign}\left[\sum_i \alpha_i y_i K_a(\tilde{x}, \tilde{x}_i) + b\right] \qquad (15)$$

Here, of course, we don't have $K_a(\tilde{x}, \tilde{x}_i)$ in closed form. This apparent obstacle can be overcome by using (3) and (11) to obtain (see [7] for details)

$$f_a(\tilde{x}) \;=\; \text{sign}\left[\sum_i \alpha_i y_i K(\tilde{x}, \tilde{x}_i) - \sum_i \alpha_i y_i < a, x_i > + b\right] \qquad (16)$$

Denote by

$$h_i \;= < a, x_i > \quad \text{for } i = 1, \ldots, n, \;\; \bar{h} = [h_1 \; h_2 \; \ldots h_n]^T \qquad (17)$$

By (12, 13) the values h_i are

$$h_i \;= < \sum_{j=1}^{n} \gamma_j x_j, x_i > \;=\; \sum_{j=1}^{n} \gamma_j K(\tilde{x}_j, \tilde{x}_i) \qquad (18)$$

Hence, a shift in the origin amounts to a constant correction term in the classifier, having the value

$$\Delta b \;=\; \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i y_i \gamma_j K(\tilde{x}_i, \tilde{x}_j) \qquad (19)$$

Note that this quantity is in general non-zero for $\gamma_i = 1/n$, i.e in the case of the "standard" centering method. Moreover, if the origin is initially far away from the data, the value of Δb can be quite large.

Equation (16) has a geometric interpretation illustrated in figure 2. This result is a direct consequence of the fact that the optimal hyperplane used by the SVM to classify the

[2] Found at www.stat.washington.edu/mmp

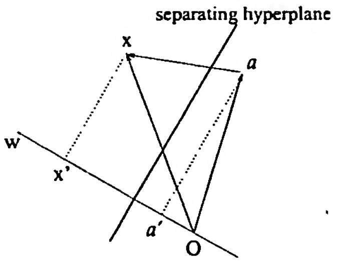

Figure 2: The geometry of origin shift and its effect on b. O, a, x, are respectively the old origin, new origin and a data point; x', a' are the projection of x, a on the normal w to the separating hyperplane (this direction is invariant to translation). The classification threshold b is the distance between the origin (old or new) to the hyperplane, and the change Δb due to the change in origin equals Oa' the projection of Oa on w. The correction Δb accounts for the fact that the origin was shifted during training while the new data are classified with the original, unshifted, kernel K.

new points is *invariant to translations in feature space.* In other words, after shifting the origin we obtain the same classifier as we would from the original kernel with better numeric stability. The SVM classification with centered data can then be summarized as follows:

1. **Preprocessing:**
   (a) Compute the Gram matrix K using definition (7)
   (b) Compute the centered Gram matrix K_a by (9)
   (c) Compute the scalar products h_i using (18)
2. **Training:** Call SVM-SOLVER(K_a, y). This outputs b, α_i, $i = 1, \ldots n$.
3. **Postprocessing:** $b \leftarrow b - \Delta b$
4. **Classification:** Classify any new data points just as for an unmodified SVM classifier, i.e using b, α_i, $i = 1, \ldots n$ and the support vectors according to (8).

In conclusion, centering in feature space only adds extra work in the SVM training phase, being essentially transparent in the classification phase.

5 A general centering method

We have shown how to perform an origin shift that optimizes the criterion (10). Now we proceed to generalize this method to optimizing any criterion $J(K_a)$ that is a function of the Gram matrix only. Examples of such criteria are

- Minimizing the sum of the cosines between all pairs of examples in different classes

$$\sum_{y_i=1}\sum_{y_j=-1} \cos \angle(x_i, x_j) \qquad (20)$$

  Since the cosine between two points in feature space is a valid kernel (which places all the data points on the unit sphere), this criterion is equivalent to simple centering in a different feature space. Note however that the relationship between the two feature spaces is not straightforward.

- Maximizing the *kernel alignment* of [6] defined as

$$A(K_a) = \frac{y^T K_a y}{n|K_a|_F} \qquad (21)$$

  with $|K_a|_F$ being the Frobenius norm of K_a.

- Another apparently useful criterion is maximizing the "unnormalized" alignment

$$y^T K y \qquad (22)$$

  At a closer inspection however, it can be seen that, unless $n^+ = n^- = n/2$ this criterion has a maximum for $a \rightarrow \infty$ in any direction so we do not recommend its usage.

Let the centering criterion be

$$\max_a J(K_a) \qquad (23)$$

We assume that J is a suitably smooth function of elements the Gram matrix, and in particular that its gradient is well defined. We show how to optimize J by gradient ascent.

For this purpose, we first need to compute the gradient of J with respect to a.

$$\nabla_a J(K_a) = \sum_{i=1}^{n}\sum_{j=1}^{n} \frac{\partial J}{\partial K_a(\tilde{x}_i, \tilde{x}_j)} \nabla_a K_a(\tilde{x}_i, \tilde{x}_j) \quad (24)$$

From equation (11)

$$\nabla_a K_a(\tilde{x}_i, \tilde{x}_j) = -x_i - x_j + 2a \qquad (25)$$

The gradient is a vector in feature space and, by combining the two above formulae, one easily sees that the gradient is a linear combination of a and the data vectors. Taking a step in the direction $\nabla_a J$ with step size η means

$$a \leftarrow a + \eta \nabla_a J \qquad (26)$$

The step $\delta_a = \eta \nabla_a J$ is itself a linear combination of a and the data vectors, hence

$$\delta_a = \gamma_0 a + \sum_i \gamma_i x_i \qquad (27)$$

with

$$\gamma_0 = -\sum_{i=1}^{n} \gamma_i, \quad \gamma_i = -2\eta \sum_{j=1}^{n} \frac{\partial J}{\partial K_a(\tilde{x}_i, \tilde{x}_j)}, \quad \text{for } i = 1, \ldots n \qquad (28)$$

All the coefficients γ above can be easily computed using only kernel evaluations. At each step of the iteration, we update: the Gram matrix K_a, the scalar products h_i =<

$a, x_i >$, $i = 1, \ldots n$ and the square length of a, $h_0 =< a, a >$ as follows:

$$< x_i, x_j >_{a+\delta_a} = < x_i, x_j >_a - < x_i, \delta_a > - < x_j, \delta_a > + 2 < \delta_a, a > + < \delta_a, \delta_a > \tag{29}$$

$$< \delta_a, x_i > = \gamma_0 < x_i, a > + \sum_{i'} \gamma_{i'} < x_i, x_{i'} > \tag{30}$$

$$< \delta_a, a > = \gamma_0 < a, a > + \sum_{i} \gamma_i < a, x_i > \tag{31}$$

$$< \delta_a, \delta_a > = \sum_{i'j'} \gamma_{i'} \gamma_{j'} < x_{i'}, x_{j'} > + 2 \sum_{i'} \gamma_{i'} < x_{i'}, a > + \gamma_0^2 < a, a > \tag{32}$$

$$< a + \delta_a, a + \delta_a > = < a, a > + 2 < a, \delta_a > + < \delta_a, \delta_a >$$

$$< a + \delta_a, x_i > = < a, x_i > + < \delta_a, x_i > \tag{33}$$

With the previous notation for $\overline{\gamma}$ and Γ we can summarize the gradient ascent algorithm as follows

1. Initialize $K_a = K$, $\overline{h} = 0$, $h_0 = 0$.
2. Compute γ_i for $i = 0, \ldots n$ by (28)
3. Update K_a, $\overline{h}$ and h_0 by (29-33)
4. Go to step 2 until convergence

From the computational point of view, each gradient step requires order n^2 computations: order n^2 derivative evaluations in step 2 and order n^2 update operations in step 3. This is of the same order of growth with one whole evaluation of the Gram matrix and affects only the training phase of the SVM classification.

For large data sets, evaluating the whole K matrix is prohibitive and state-of-the-art SVM implementations evaluate only a subset of rows of K. In that case, the centering algorithms presented here would be prohibitive as well. We can easily fix this problem by using only a subsample of the data for centering, in a way similar to [12, 14]. For the simple centering method, we would sample e.g. $n'/2$ data points from each class and represent a as the arithmetic mean of the subsample. This would still ensure that the new position of the origin falls inside the convex hull of the data, but the extra amount of computation per row of K_a will be of order n'^2.

We can also use sampling to reduce the computational complexity of the general centering method. In this case the solution is to redefine the optimality criterion J to involve only a submatrix of K_a, depending on a subset of $n' << n$ points. While the solution may not work for any criterion, it is a reasonable approximation in the case of e.g optimizing the kernel alignment [12].

6 Experiments

6.1 Shifting and recentering in feature space

In the experiments we used the SVMLIB [2] source code, modified in order to accept a user-defined Gram matrix.

The first set of experiments is performed on artificial data and aims to show that (1) drastic origin shifts in feature space harm the performance of an SVM classifier and (2), that the simple centering algorithm is able to restore the effects of the shift. We generated data normally distributed around two concentric circles as in figure 1 and computed its Gram matrix K. Then we shifted the data in a random direction in feature space by a predetermined distance $|a|$ and computed the "shifted" Gram matrix K_s. We then centered the shifted data by the simple centering method described in section 3 and computed the "centered" Gram matrix K_c. Finally we trained an SVM using each of the three Gram matrices and evaluated it on test data from the same distribution.

The experiment was repeated 10 times with different samples and shift directions for every value of $|a|$. We used the degree 2 polynomial kernel $K(\tilde{x}, \tilde{z}) = (1 + \tilde{x}^T \tilde{z})^2$ and the RBF kernel $K(\tilde{x}, \tilde{z}) = e^{-(\tilde{x}-\tilde{z})^2/\sigma^2}$. The training (test) set size was 300 (200) in all cases. The results are shown in figure 3.

The second set of experiments was similar to the first, except that now we used real data sets from the UCI repository. The data set sizes are given in table 1. The shift length was 1000 for all data sets. For each data set, the experiment was run 10 times with different randomly sampled training sets. The results are shown in figure 4.

From the two experiments we see first that a large shift is detrimental to classification performance. The recentered and original classifiers are almost identical for all the artificial data experiments and for all but one of the real data sets (wdbc). This shows that recentering has indeed a restorative effect on data placed far away from the origin in feature space. The figures also show the effect on shifting and recentering on kernel alignment: the alignment of the shifted kernel is practically 0 for the artificial data and drastically reduced for the real data. Recentering brings alignment back to near or above the original values. The number of support vectors, an indirect indicator of generalization performance, grows with the size of the shift in the polynomial kernel and for all but the largest shift in the RBF kernel, but drops elsewhere. The drop is very likely an artefact of the SVM-SOLVER software for extremely ill-posed problems (note that a shift of size 1000 is extremely large in the case of the RBF kernel).

In the third set of experiments, we compared the centering method described in section 3 with shifting the origin in the center of weight of the data. To maximize the difference between the two methods, in these experiments the number of examples in one class was 20 times larger than the number of examples in the other class. Testing was done on data generated from the same distribution as the training data. The experimental setup mimicked the one for the first experiment.

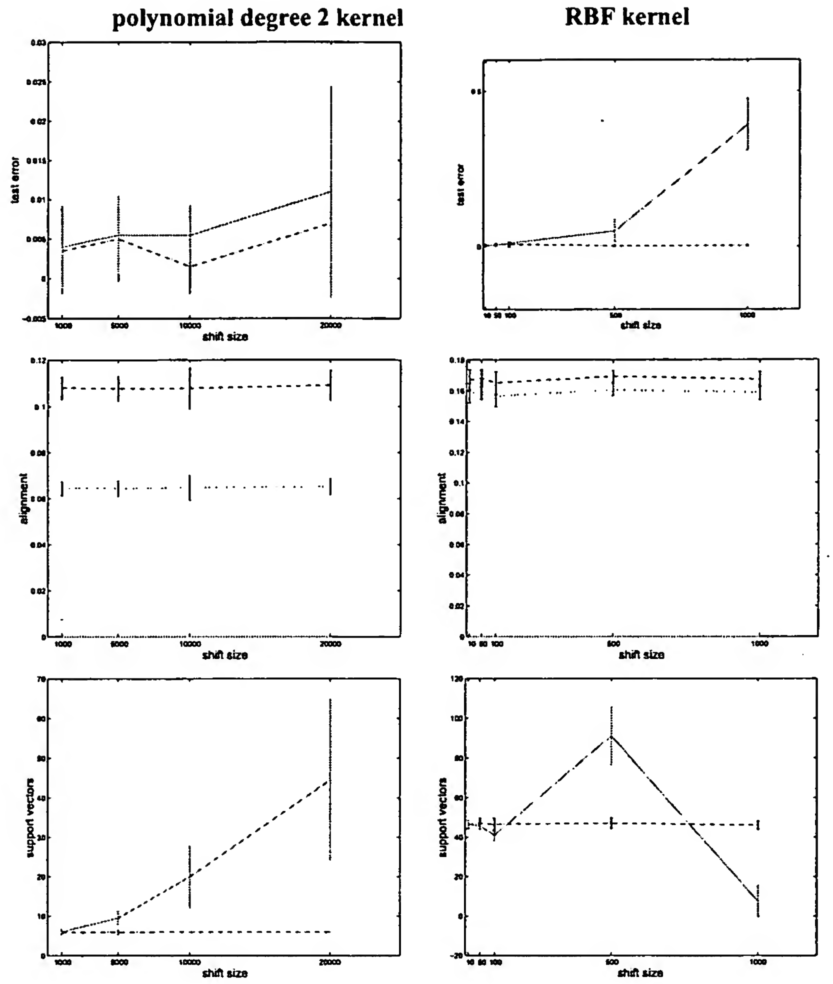

Figure 3: SVM classification with original (dotted line), shifted (full line) and recentered (dashed line) data for different values of the shift length $|a|$. The original data are generated from the distribution shown in figure 1 (two concentric circles). These data are shifted in a random direction by an amount $|a|$ in feature space to obtain the shifted data. The shifted data are then recentered as in section 3 to obtain the recentered data. The figures depict the test error, kernel alignment and number of support vectors for the resulting SVMs, in the case of the polynomial degree 2 kernel (left) and of the RBF kernel (right). Results are averaged over 10 randomly sampled training sets of size 300. The original and centered SVMs are identical in all cases. The alignment of the shifted kernel is practically 0.

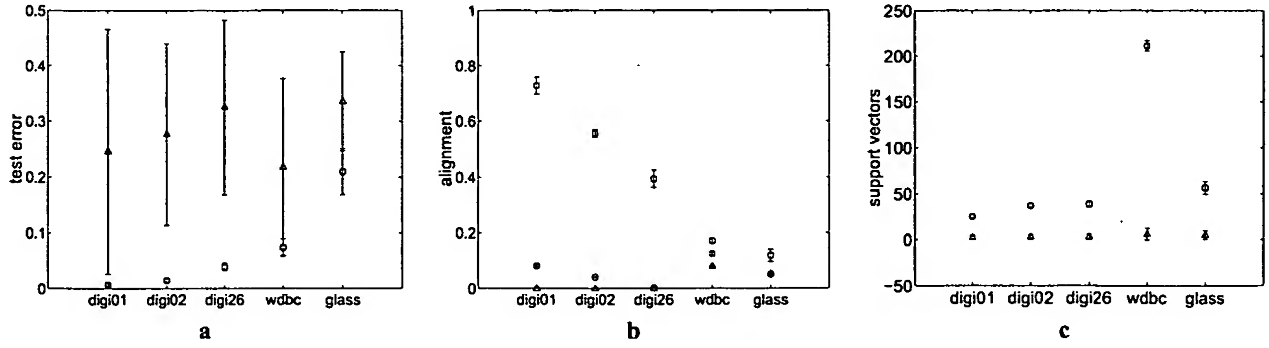

Figure 4: Shifting and recentering on real data sets: (a) test error, (b) alignment, (c) number support vectors. Circles represent original data, triangles – shifted data, squares – recentered data. The data sets are described in table 1. Each experiment was repeated 10 times with random direction shifts. The shift lenght $|a|$ was 1000 and the kernel was the RBF kernel in all cases. Note that the original and centered results are superimposed in **a**, **c**.

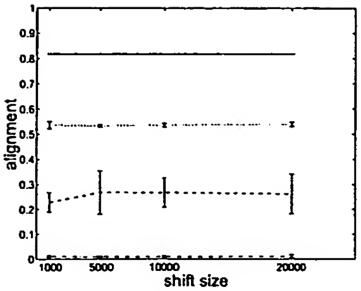

Figure 5: Alignment of the Gram matrices vs shift value for 4 kernels: original (dotted), shifted (full), recentered by the center of weight method (dash-dot) and recentered by the simple method of section 3.

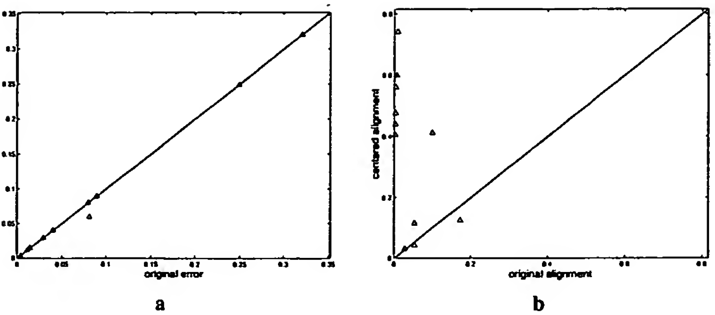

Figure 6: SVM classification of original versus centered data in 12 experiments with 10 data sets: **(a)** test error, **(b)** kernel alignment. Each data point is the average of 10 random training/test splits. The data sets are described in table 1.

For both centering methods, the recentered and the original classifier are essentially the same for the whole range of shifts. (The detailed results are in [7]). Therefore we can safely conclude that there is no practical difference between the two centering methods.

An interesting aspect is revealed by the alignment plots in figure 5. Unlike figure 3 the alignment is maximum for the *shifted* data, while centering drastically *reduces* it. A quick analysis reveals the cause of this behavior: for a sufficiently large origin shift in feature space, the value of the alignment tends to

$$A(K_a) \longrightarrow (n^+ - n^-)^2/n^2 \qquad (34)$$

In our case, n^+ is 20 times larger than n^- which yields the value $A(K_s) = 0.82$, in perfect agreement with the experiments. This strongly cautions us that optimizing the kernel alignment may not always produce the best classifier.

6.2 Centering real data

In this set of experiments, we applied the simple centering algorithm to real data. We computed the Gram matrices before and after centering, denoted by K and K_a respectively), trained an SVM for each of them, and evaluated its performance on an independent test data set. The data sets, training and test set sizes, kernel types and parameters are given in table 1. The SVM parameters were chosen so as to produce reasonable but not necessarily optimal classification results on the original data. This was done before the centering experiments, with one random training/validation split of the original data.

The results are summarized in figure 6. Each point in the figure represents the average of 10 random training/test splits. The test error plot shows that, as expected, centering has no effect in most cases but it improves performance occasionally (here, in one case: the wdbc data with polynomial kernel). In none of the experiments did data centering hurt performance. In most cases where performance wasn't improved, the SVM classifiers from the centerd and original data were virtually identical.

The kernel alignment is slightly reduced in 2 of the 12 experiments and dramatically increased in 8 others. Again, we notice that improving the alignment per se does not necessarily guarantee an improvement in the classification performance.

7 Discussion

This paper has presented a family of methods for data shifting and centering in feature space. They can be used in conjunction with any kernel machine that incorporates the information from the data in a Gram matrix. Data centering in feature space does not, in theory, affect the resulting classifier. We have shown that in practice, it can have a benefic effect when the Gram matrix is ill conditioned due to a poor position of the origin w.r.t the data in feature space. We have found no instances where data centering hurt the classification performance.

When used for data centering, translation in feature space requires extra work only in the training stage of the SVM. The extra computations are of the order n^2, but can be reduced by standard sampling schemes.

There have been many previous studies on kernel adaptation [6, 1, 5]. Our centering methods differ from the previous as they do not attempt to obtain a more appropriate kernel and they do not change the geometry of the problem. The aim of data centering is merely to hand the SVM-SOLVER a problem instance with better numerical properties.

Additionally, we have shown that the "standard" centering method present in the literature requires a correction term for b. The experimetns have also illustrated interesting aspects of the (lack of) relationship between the kernel alignment and classification performance in practice. In particular, translation in feature space can greatly change the alignment with no effect on the classifier perfromance.

Table 1: The data sets used in the experiments.

Name	Description	# inputs	# train	# test	SVM parameters
cmc	Contraceptive data, UCI repository (class 1 vs all others)	9	400	523	RBF $\sigma^2 = 100,\ C = 1000$
glass	Glass data, UCI repository (class 2 vs all others)	9	130	84	poly2, RBF $\sigma^2 = 2,\ C = 1000$
wdbc	Wisconsing breast cancer data, UCI repository	30	312	257	poly2, RBF $\sigma^2 = 1000,\ C = 1000$
digi*ab*	Handwritten digits from the USPS (digit a vs digit b where $a, b \in \{0, 1, 2, 6\}$)	64	200	400	RBF $\sigma^2 = 1000,\ C = 1000$

We therefore experimented with factoring out the effects of origin translation by first centering the data and then maximizing the alignment. The results are in the full paper.

Here, the results of the theoretical investigation into data translation in feature space have been used solely for data centering. We envisage however a more interesting realm of applications: shifting the data in order to obtain new kernels, parametrized by the shift. Obtaining a new kernel by origin shifts is possible with composite kernel, such as the ones used in the classification of string data (see e.g [13]). If one shifts the element kernels before composition, then the operation amounts to more than a translation at the leverl of the composite kernel and it does affect the problem geometry. Preliminary experiments in this direction are already under way.

Acknowledgments

The author thanks Deepak Verma for discussions and some help with the code, Chris Watkins for discussions on string kernels and Chih-Chung Chang and Chih-Jen Lin for writing and making available the **LIBSVM** software.

References

[1] S. Amari and S. Wu. Improving support vector machines by modifying kernel functions. *Neural Networks*, pages 783–789, 1999.

[2] C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[3] C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20:273 – 297, 1995.

[4] N. Cristianini. Support vector and kernel machines. Tutorial at ICML, 2001.

[5] N. Cristianini, C. Campbell, and J. Shawe-Taylor. Dynamically adapting kernels in support vector machines. NeuroCOLT Technical Report NC-TR-98-017, Royal Holloway College, University of London, UK, 1998.

[6] N. Cristianini, J. Shawe-Taylor, A. Elisseeff, and J. Kandola. On kernel target alignment. In T. Dietterich, S. Becker, and D. Cohn, editors, *Neural Information Processing Systems*, number 14, Cambridge, MA, 2002. MIT Press.

[7] M. Meilă. Data centering in feature space. Technical report, University of Washington, 2002.

[8] B. Schölkopf. Statistical learning and kernel methods. Technical Report MSR-TR-2000-23, Microsoft Research, Cambridge, UK, 2000.

[9] B. Schölkopf, J. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson. Estimating the support of a high-dimensional distribution. Technical Report 99-87, Microsoft Research, 1999. To appear in *Neural Computation*, 2001.

[10] B. Schölkopf, A. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319, 1998. Technical Report No. 44, 1996, Max Planck Institut für biologische Kybernetik, Tübingen.

[11] B. Schölkopf, A. Smola, R. Williamson, and P. L. Bartlett. New support vector algorithms. NeuroCOLT Technical Report NC-TR-98-031, Royal Holloway College, University of London, UK, 1998. Published in *Neural Computation* 12(5):1207–1245, 2000.

[12] J. Shawe-Taylor, N. Cristianini, and J. Kandola. On the concentration of spectral properties. In S. B. Tom Dietterich and D. Cohn, editors, *Neural Information Processing Systems*, number 14, Cambridge, MA, 2002. MIT Press.

[13] C. J. C. H. Watkins. Dynamic alignment kernels. Technical Report CSD-TR-98-11, Royal Holloway, University of London, 1999.

[14] C. Williams and M. Seeger. Using the nyström method to speed up kernel machines. In *International Conference on Machine Learning*, number 17, 2000.